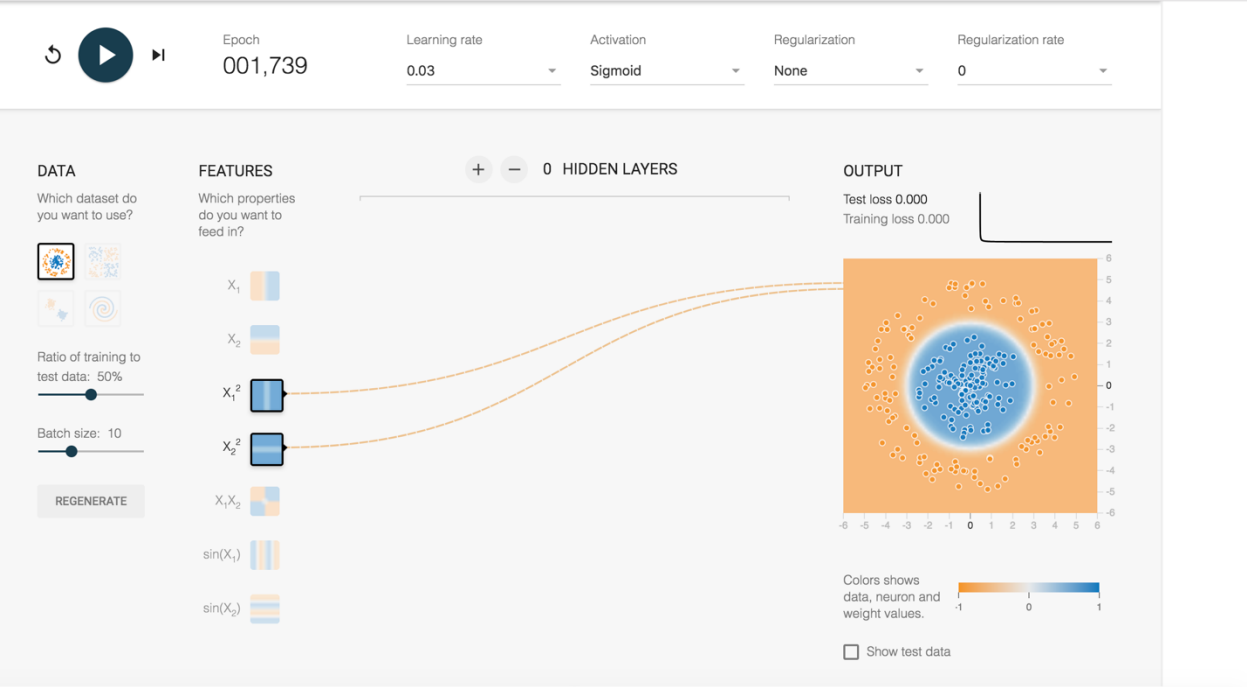
HW4 – CS 4501 Machine Learning

Roman Sharykin
Collaborated with Jed Barson

1. Neural Network Playground

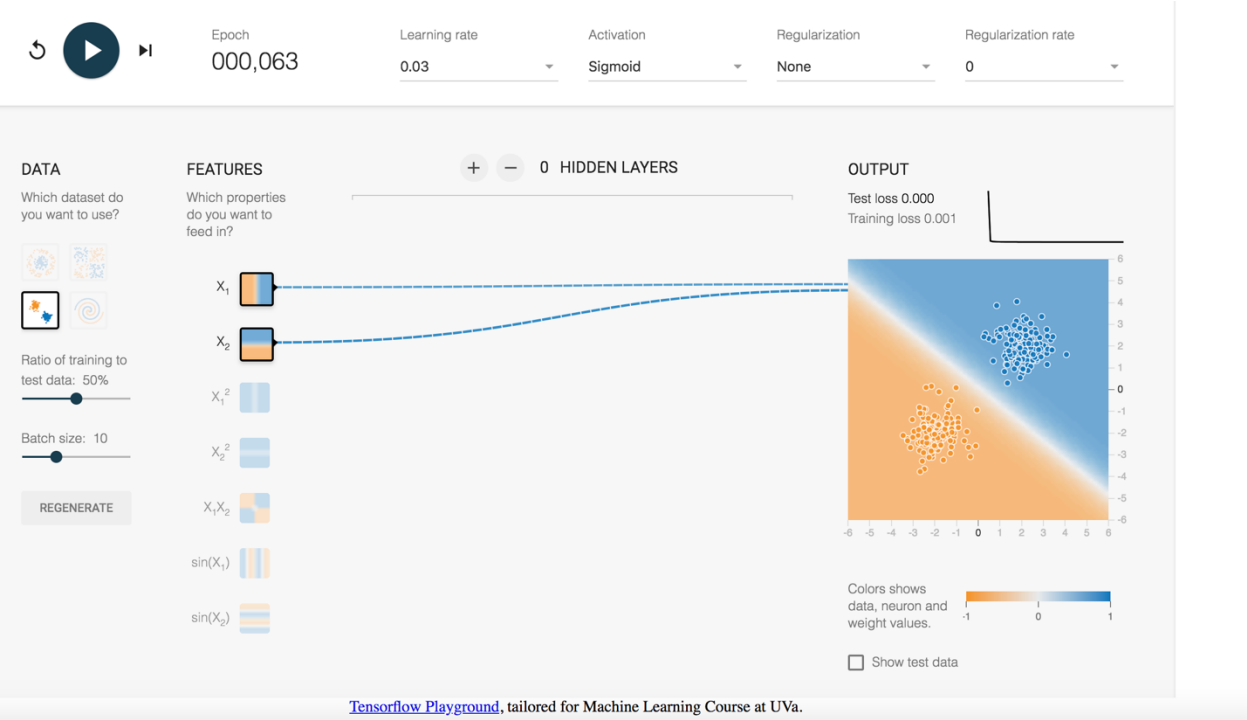
1.1 Hand-crafted Feature Engineering

Circle:



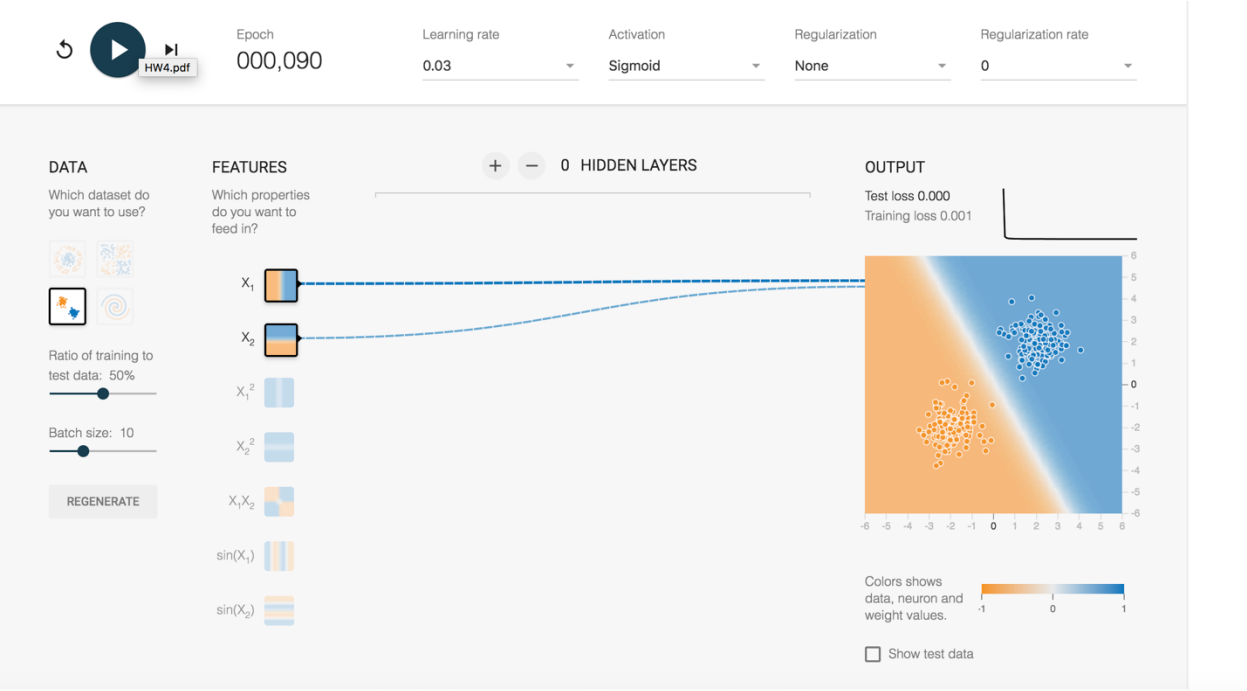
Features: 2 (X1^2, X2^2)
Epochs: 1,739
Test loss: 0.000
This configuration works because combining the X1^2 and X2^2 features, instead of getting separated columns or rows like you would for only one of these features, you get a circle in the middle which matches our dataset.

Exclusive Or:



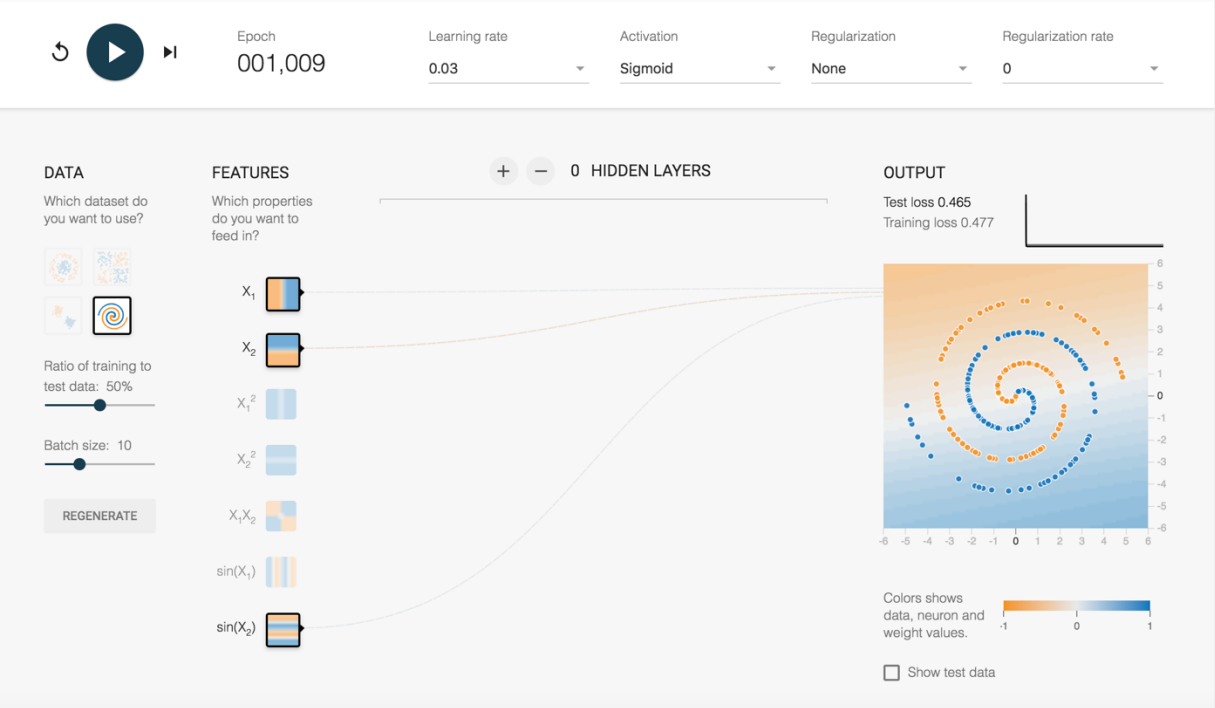
Features: 1 (X1X2)
Epochs: 63
Test loss: 0.000
This configuration works because the X1X2 feature alone matches the Exclusive Or dataset very well.

Gaussian:



Features: 2 (X1, X2)
Epochs: 90
Test loss: 0.000
This configuration works because the X1 and X2 features work together to create a diagonal boundary – alone, these features create a horizontal or vertical boundary, but together the diagonal is created which matches our dataset.

Spiral:



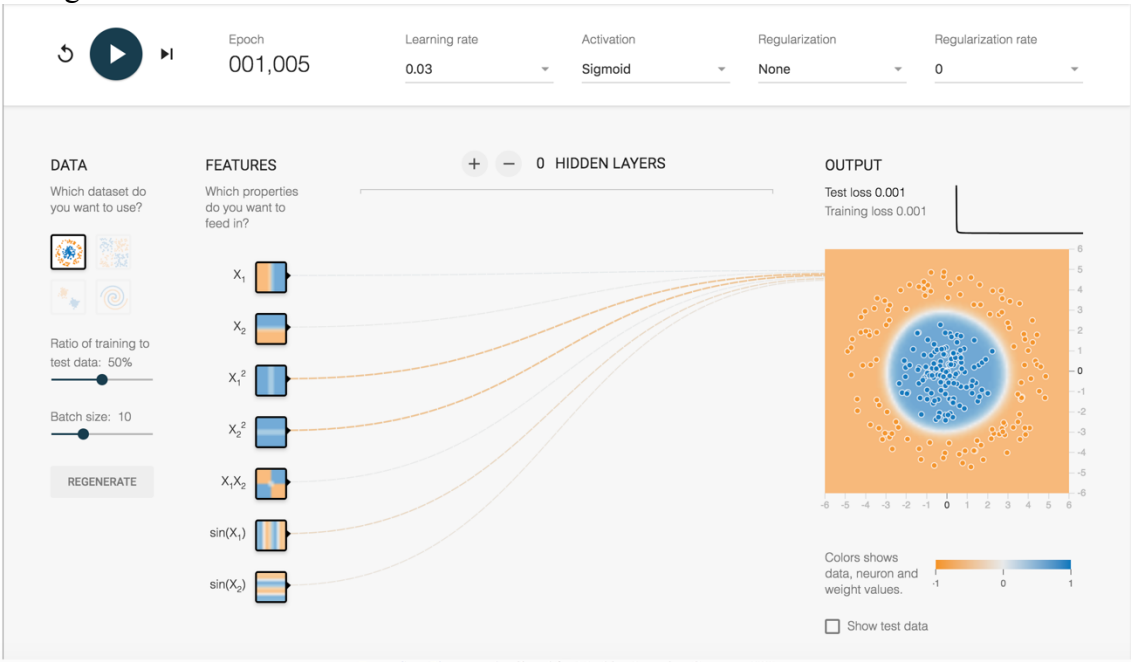
Features: 4 (X1, X2, sin(X2))
Epochs: 1,009
Test loss: 0.465.
This configuration worked better than the others I tried for this, probably because I introduced a sinusoidal function into a configuration with the two features X1 and X2. However, the test loss is still high since we weren't allowed any hidden layers, non-sigmoid activations, or regularizations, which I believe would have helped to decrease test loss here, but this is the best I could come up with using only a simple perceptron.

1.2 Regularization

Task A:

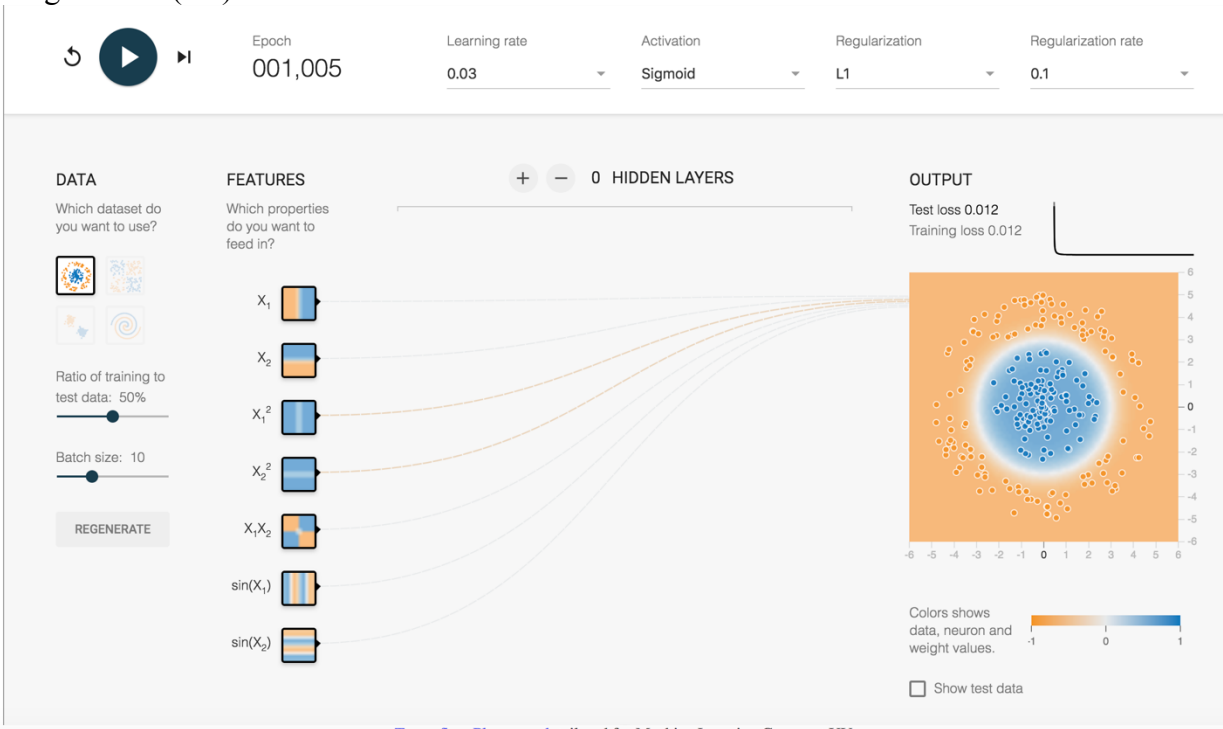
Circle:

No regularization:



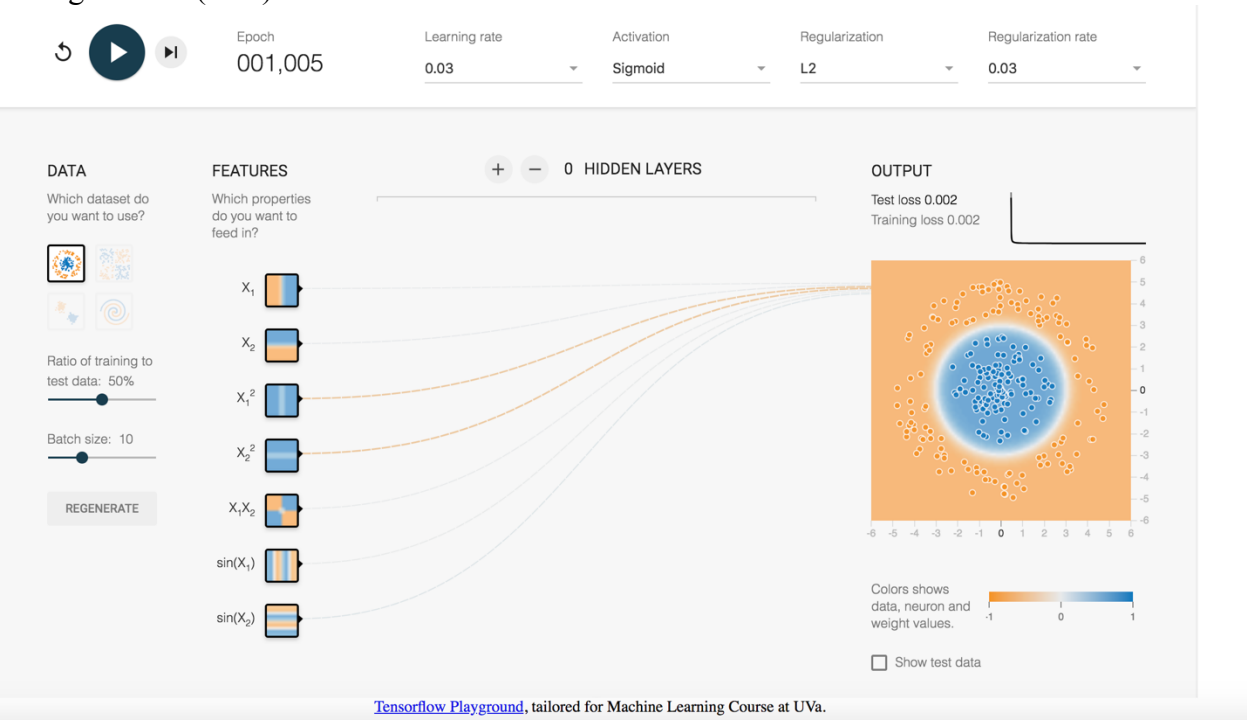
Tensorflow Playground, tailored for Machine Learning Course at UVa.

L1 regularized (0.1)



Tensorflow Playground, tailored for Machine Learning Course at UVa.

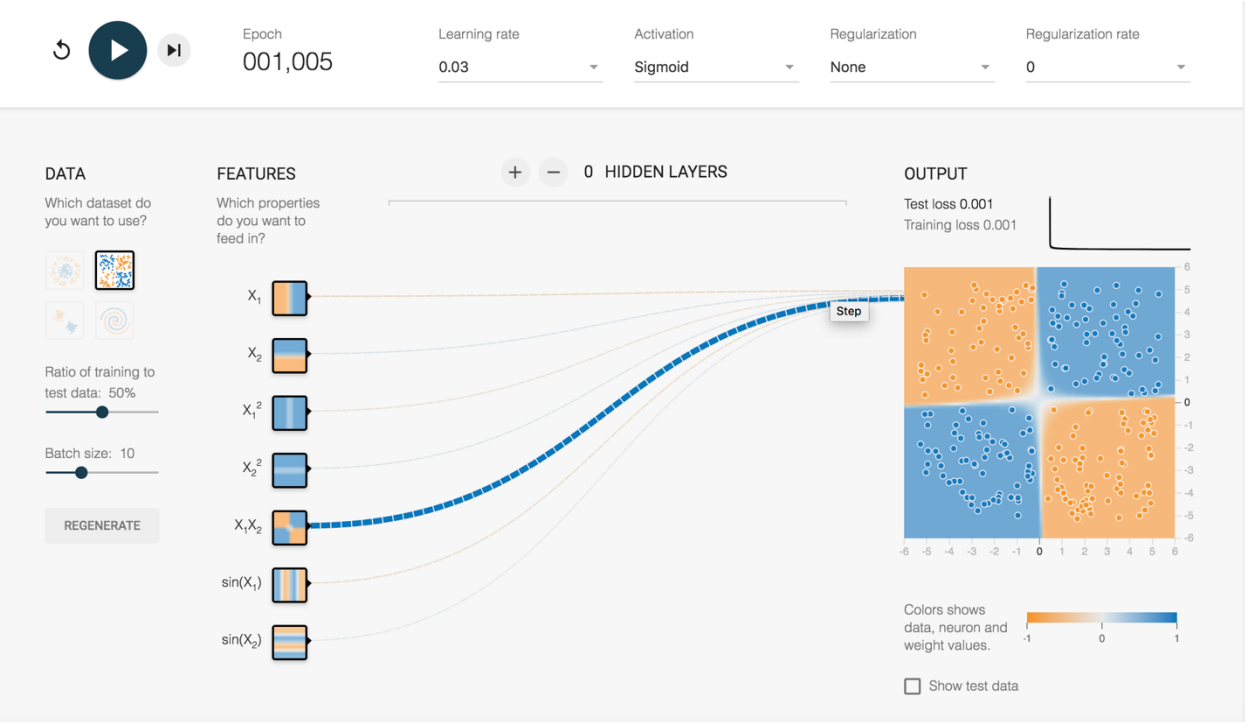
L2 regularized (0.03)



As expected, there is the least amount of test loss with no regularizations, but the boundaries are smoother when you add in L1/L2 regularizations – you can see the circle without regularization is not very uniform in shape, and we can attribute these differences due to the tradeoff with loss involved when introducing regularizations.

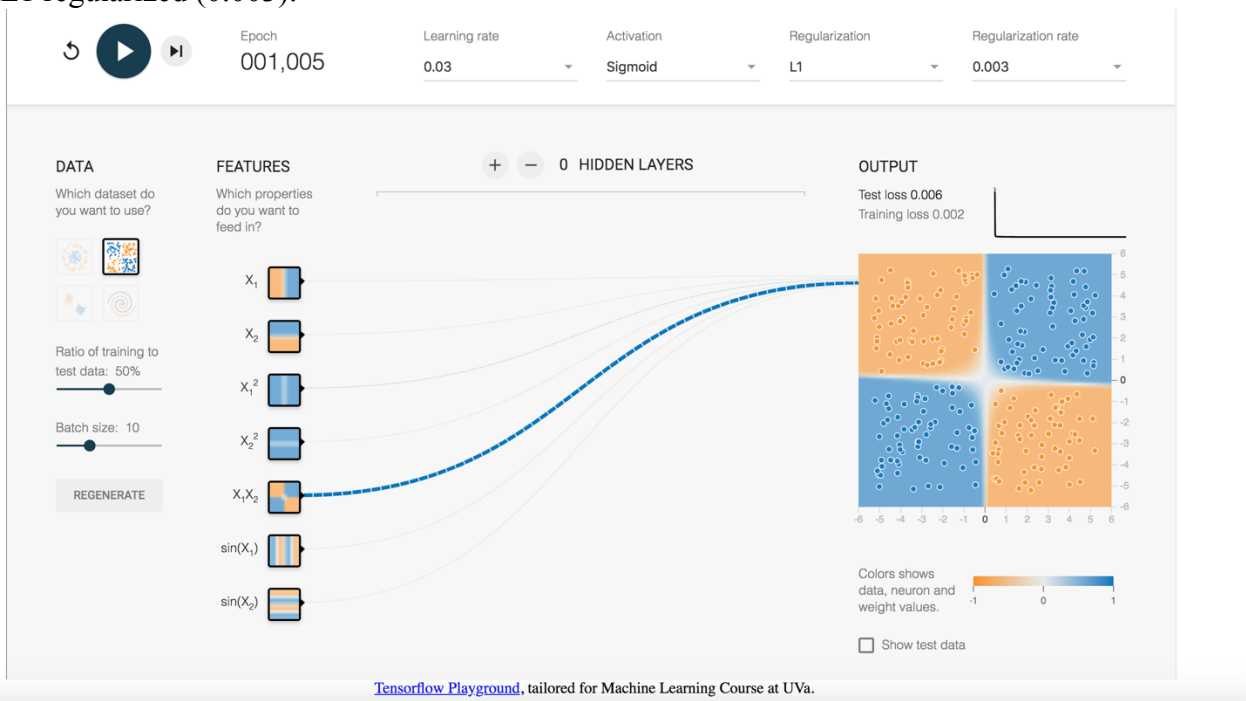
Exclusive Or:

No regularization:



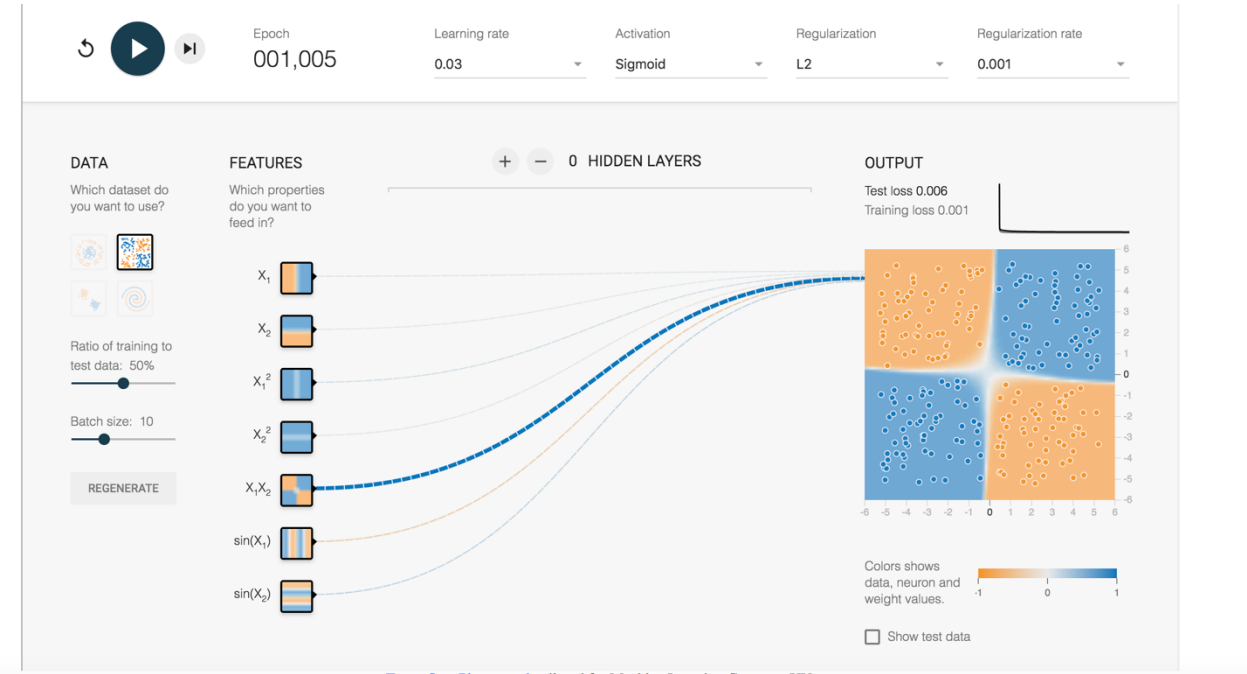
Tensorflow Playground, tailored for Machine Learning Course at UVa.

L1 regularized (0.003):



Tensorflow Playground, tailored for Machine Learning Course at UVa.

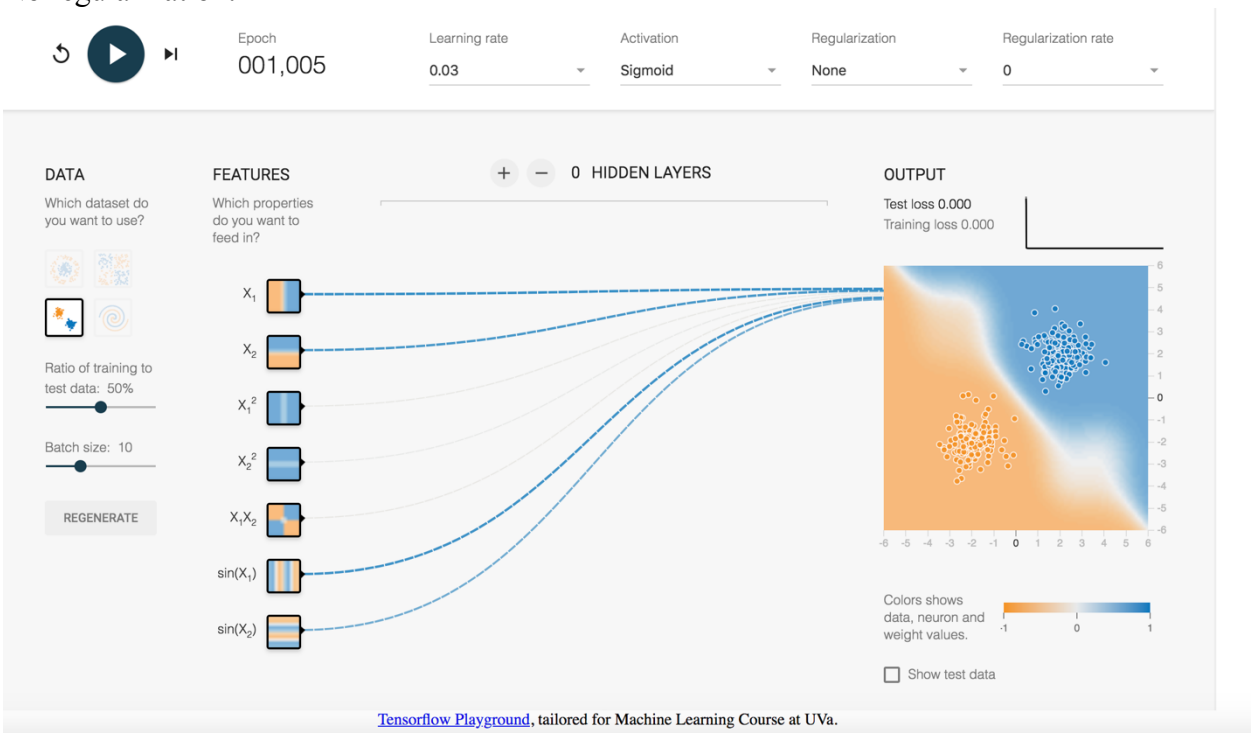
L2 regularized (0.001):



The test loss is 0.001 unregularized, but 0.006 with each of the L1 and L2 regularizations. These are the best configurations because there is a fairly low test error while also utilizing the smoothing regularization techniques. Both the regularized boundaries appear smoother, but L1 especially has a much smoother boundary which explains the loss tradeoff.

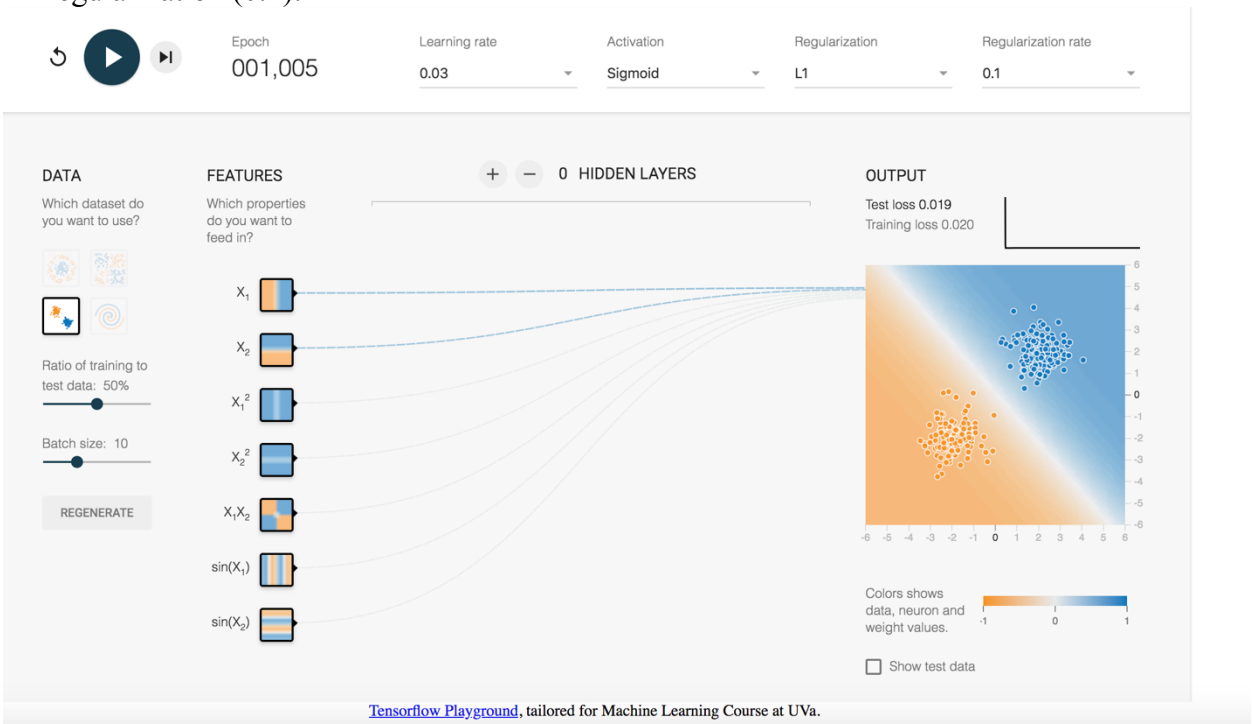
Gaussian:

No regularization:



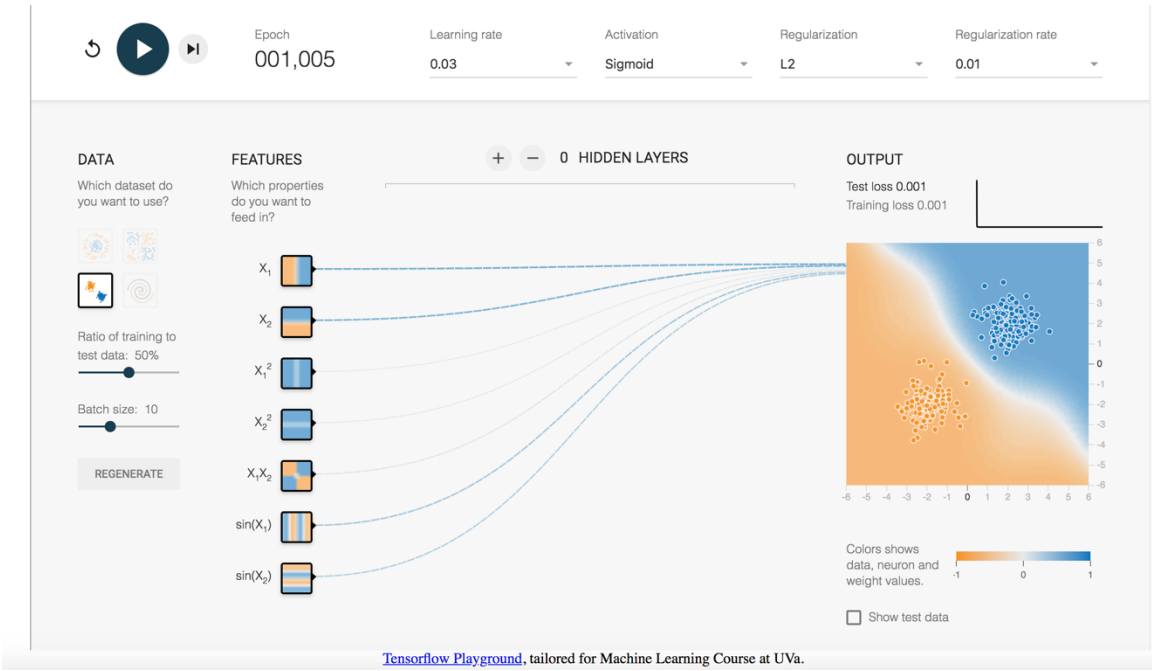
Tensorflow Playground, tailored for Machine Learning Course at UVa.

L1 regularization (0.1):



Tensorflow Playground, tailored for Machine Learning Course at UVa.

L2 regularization (0.01):

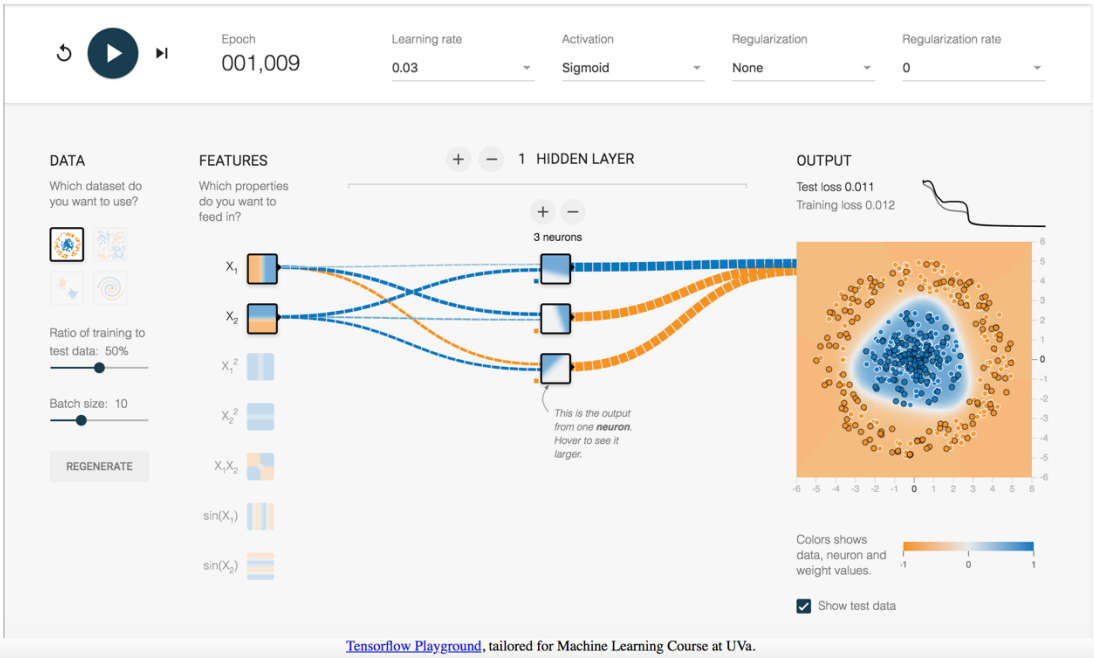


Tensorflow Playground, tailored for Machine Learning Course at UVa.

This dataset is where you can see the biggest difference between the boundary of the unregularized model and the regularized ones. There is less loss when unregularized, but as you can see the regularizations (especially L1) cause the boundaries to smooth out which is why I chose these configurations – this is the tradeoff we learned in class.

1.3 Automated Feature Engineering with Neural Network

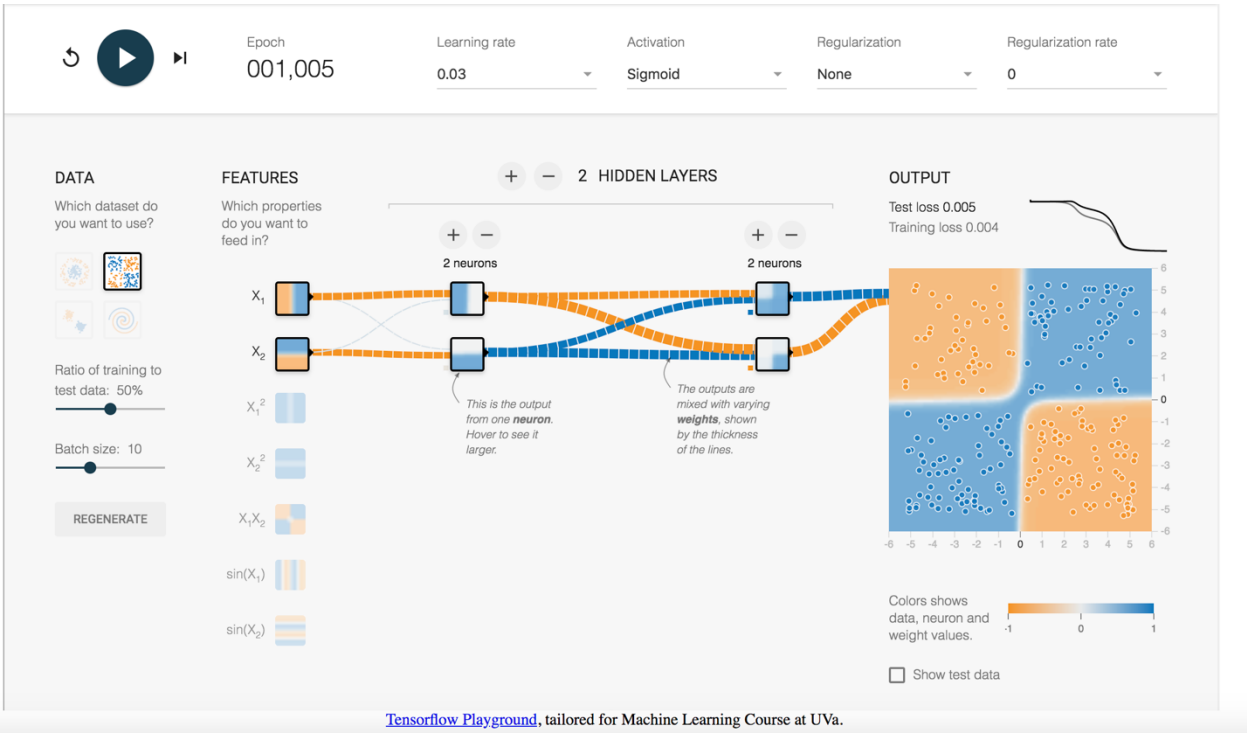
Circle:



Tensorflow Playground, tailored for Machine Learning Course at UVa.

After trying many different configurations, I was able to get a boundary that correctly separates the testing samples. I used 1 hidden layer with 3 neurons to achieve this – I ended up with 0.011 test loss after 1,009 epochs.

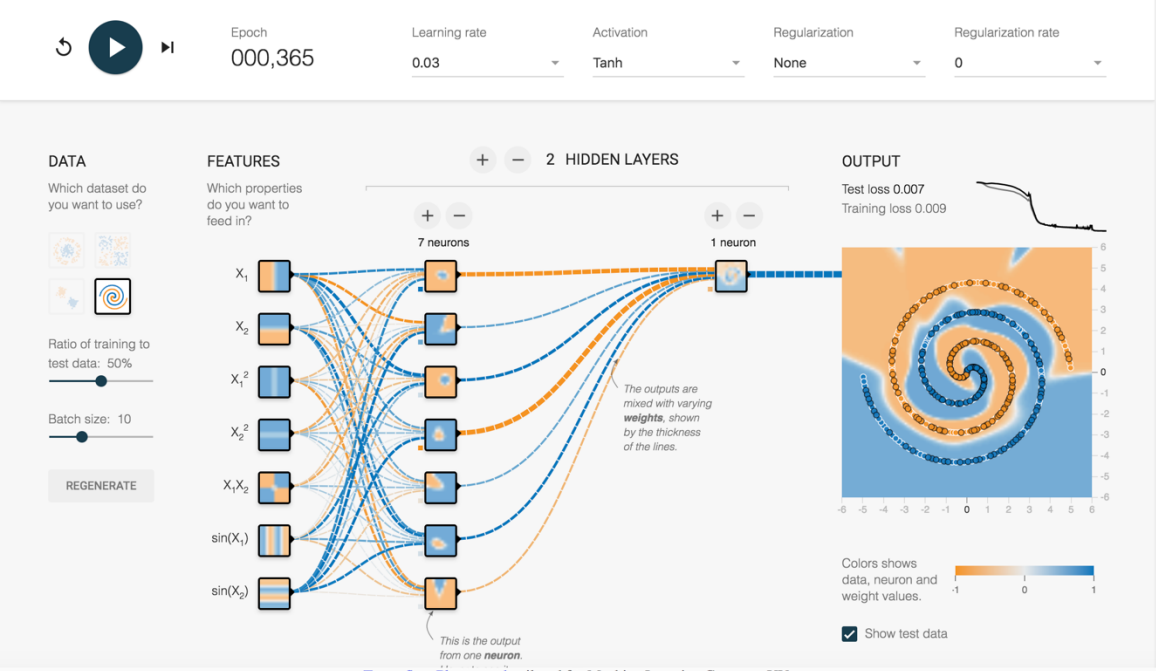
Exclusive Or:



Tensorflow Playground, tailored for Machine Learning Course at UVa.

I was able to get a boundary that correctly separates the testing samples for this dataset as well. I used two hidden layers this time, with 2 neurons in each layer – I ended up with 0.005 test loss after 1,005 epochs.

1.4 – Spiral Challenge



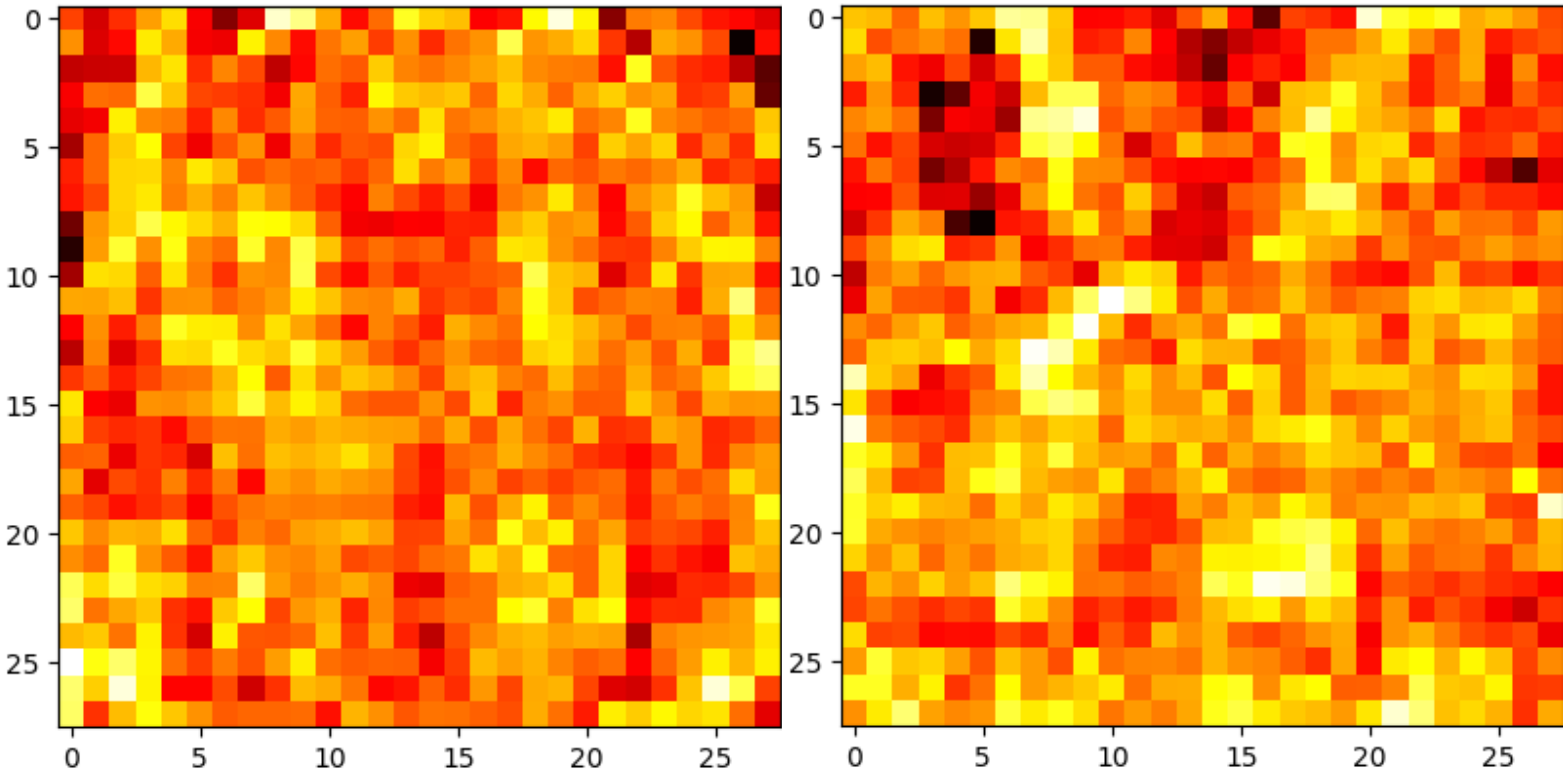
Tensorflow Playground, tailored for Machine Learning Course at UVa.

For this challenge, I realized that using tanh rather than sigmoid for the activation generally gave me much better results so I used that instead. The input features are all seven of the features, with a two-layer 7-1 neural architecture (7 neurons in layer 1, 1 neuron in layer 2), with 0.007 test loss achieved after 365 epochs.

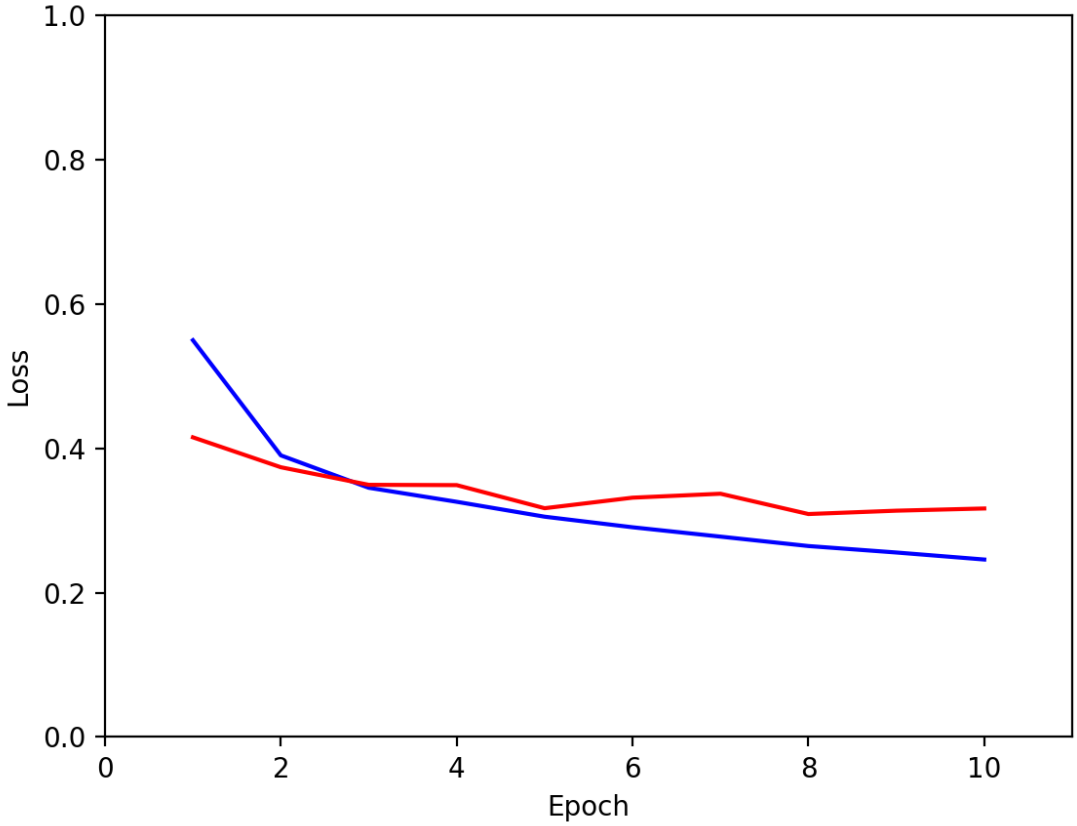
2. Deep Learning with Keras

MLP

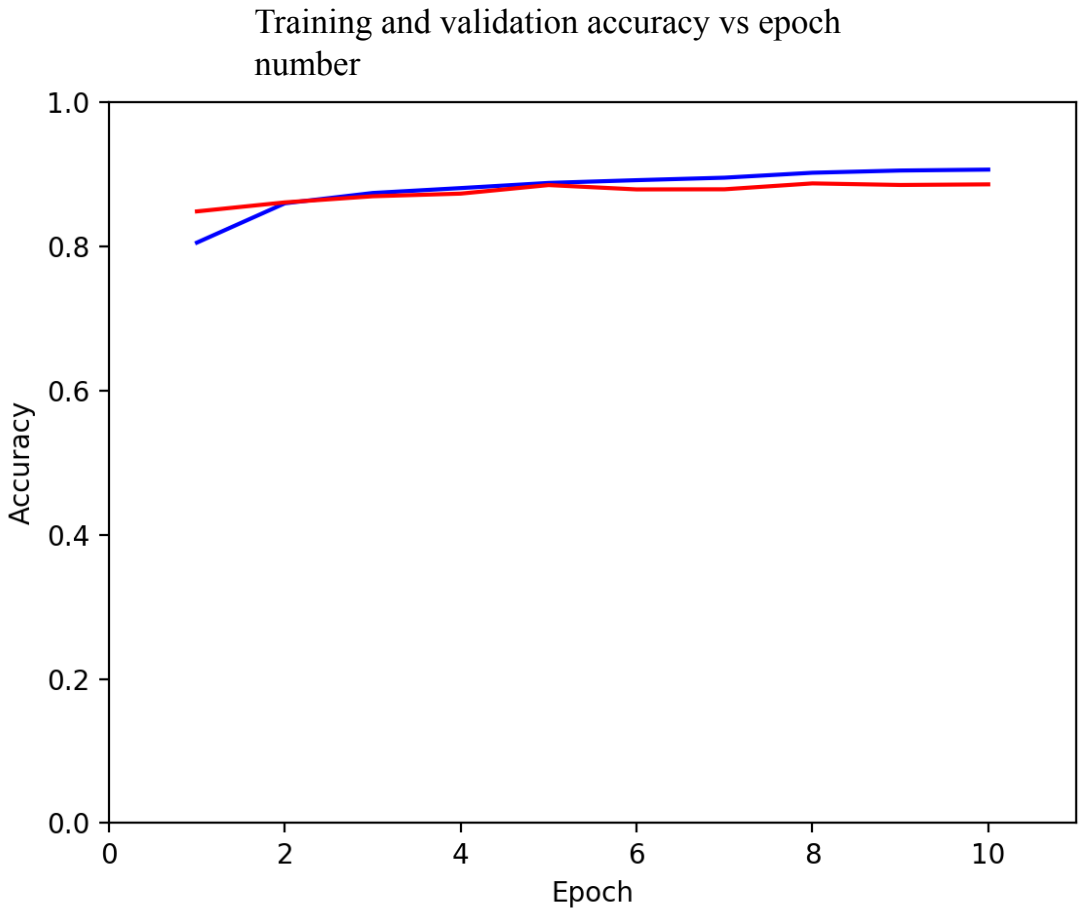
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 64)                50240
_________________________________________________________________
dense_2 (Dense)              (None, 512)               33280
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
=================================================================
Total params: 88,650
Trainable params: 88,650
Non-trainable params: 0
```



Training and validation loss vs epoch number



Train - BLUE Validation - RED



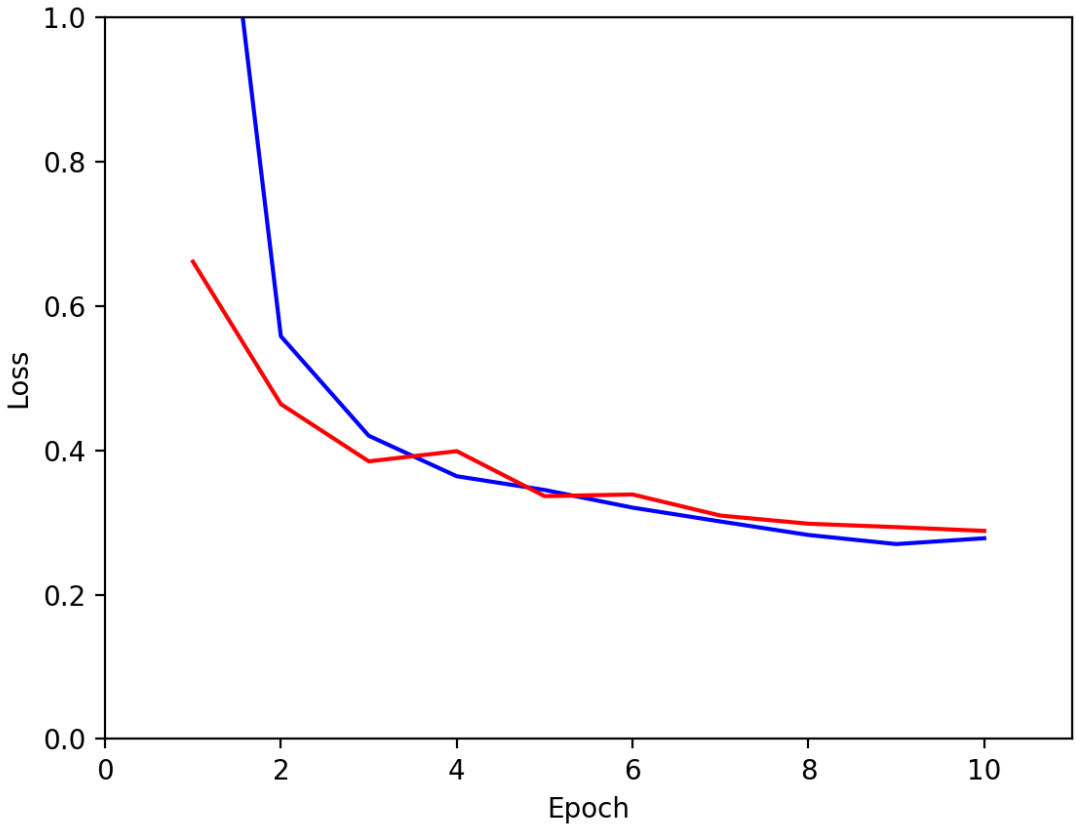
Train - BLUE Validation - RED

Cross-entropy is preferred for **classification**, while mean squared error is one of the best choices for **regression**. In classification you work with very particular set of possible output values thus MSE is badly defined (as it does not have this kind of knowledge thus penalizes errors in incompatible way).

CNN

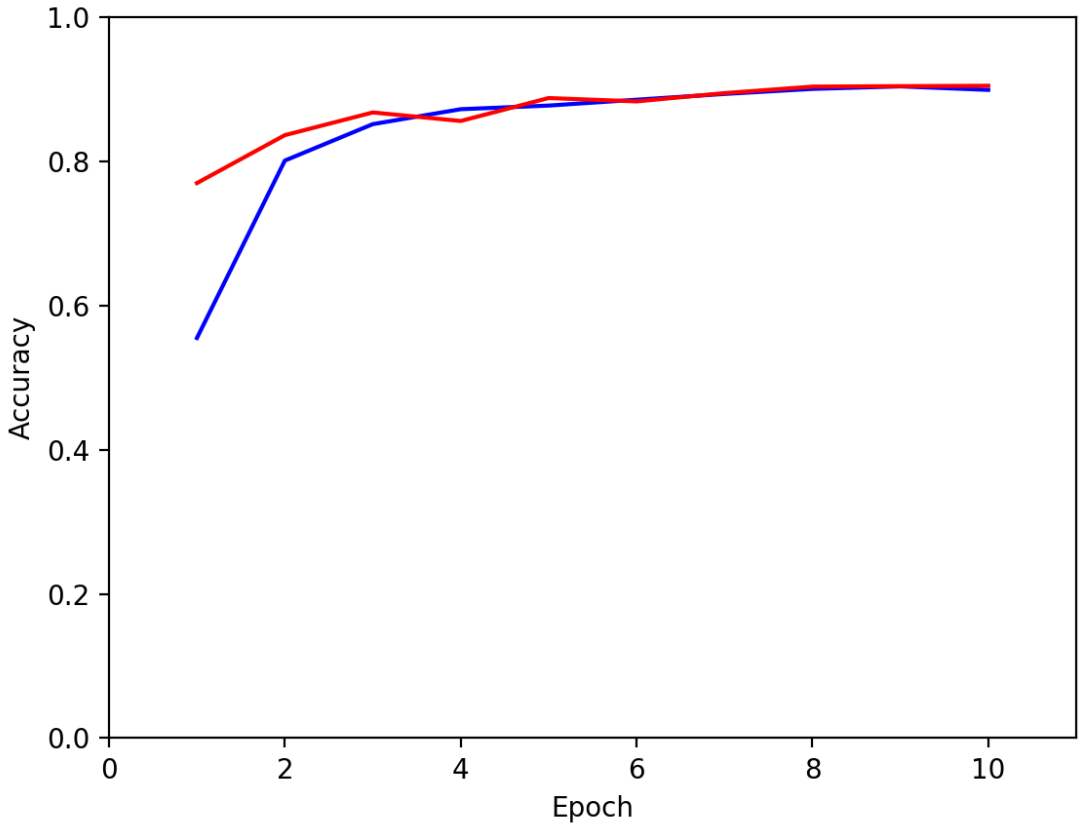
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 25, 25, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 20000)             0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                200010
=================================================================
Total params: 200,330
Trainable params: 200,330
Non-trainable params: 0
```

Training and validation loss vs epoch number



Train - BLUE Validation - RED

Training and validation accuracy vs epoch number



Train - BLUE Validation - RED

3. Sample Exam Questions

Question 1

a. For the output to be greater than 0.5, the denominator of the g(z) function has to be below 2, meaning that exp(-z) has to be below 1, meaning that z has to be a positive number for the output to be greater than 0.5. If we set the weights as w0 = -0.75, w1 = 0.5, w2 = 0.5, w3 = 1, then the only way to outweigh w0's negative value is for w3 to be used or BOTH w1 and w2 to be used (these two add up to the value of w3), otherwise z will be negative. For these reasons, these values of the weights are correct for the purposes of the question.

b.

i. True because both can be used to perform linear regression.

ii. True because error surface is affected by training data.
iii. False because there are cases where batch gradient descent works better than SGD.
iv. True because if inputs are assumed to be zero or positive like before then having a negative w0 will cause output to always be negative even if the inputs are all zero.
v. False because in the third update (delta w2), x_d2 should be squared and not doubled just as x2 is.

Question 2

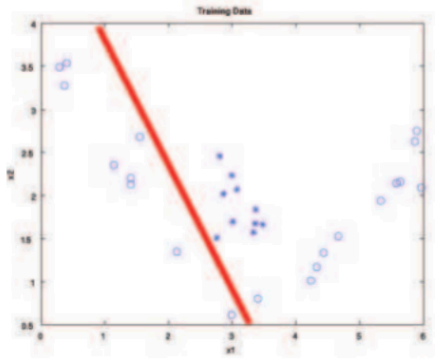
a. Assign all L's because we want linear regression so we want linear functions for all neurons.

b. The first two neurons should be L's still, but since we want a binary logistic classifier and looking at the given argmax function, the output neuron should use an S (sigmoid function).

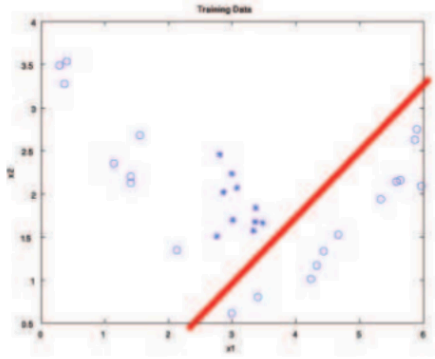
c. Since w1 and w2 go from X1, and w5 goes out of the neuron w1 goes into and w6 goes out of the neuron w2 goes into, $\beta 1 = c(w1w5 + w2w6)$. Since w3 and w4 go from X2, and w5 goes out of the neuron w3 goes into and w4 goes out of the neuron w6 goes into, $\beta 2 = c(w3w5 + w4w6)$.

Question 3

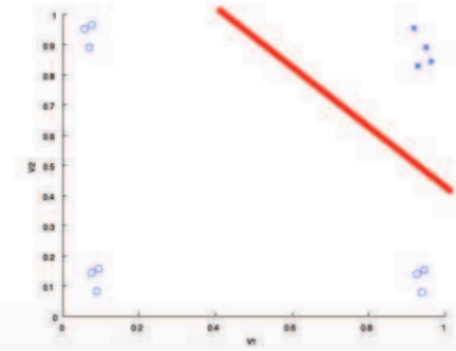
N1 (2 points)



N2 (2 points)



N3 (4 points)



The first boundary separates the left half of the “false” data points from the rest of the dataset. The second boundary separates the right half of the “false” data points from the rest of the dataset. Since the third graph is V2 vs. V1, the “true” occurrences occur when both V1 and V2 are near one, which will always be in the top right section of the graph. Therefore, we draw a boundary sectioning the “true” points in the top right from the others.